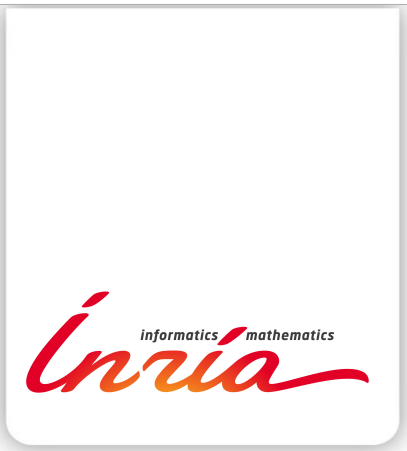
Energy-Aware Routing in Software-Defined Networks with Table Compression (using Wildcard Rules)

Frédéric Giroire, Nicolas Huin, Joanna Moulierac, Khoa Phan

RESEARCH REPORT
N° 8897
Avril 2016
Project-Team COATI

ISSN 0249-6399 ISRN INRIA/RR--8897--FR+ENG



Energy-Aware Routing in Software-Defined Networks with Table Compression (using Wildcard Rules)*†

Frédéric Giroire‡, Nicolas Huin‡, Joanna Moulierac‡, Khoa Phan§

Project-Team COATI

Research Report n° 8897 — Avril 2016 — 32 pages

Abstract: Software-defined Networks (SDN), in particular OpenFlow, is a new networking paradigm enabling innovation through network programmability. Over past few years, many applications have been built using SDN such as server load balancing, virtual-machine migration, traffic engineering and access control. In this paper, we focus on using SDN for energy-aware routing (EAR). Since traffic load has a small influence on power consumption of routers, EAR allows to put unused links into sleep mode to save energy. SDN can collect traffic matrix and then computes routing solutions satisfying QoS while being minimal in energy consumption. However, prior works on EAR have assumed that the forwarding table of OpenFlow switch can hold an infinite number of rules. In practice, this assumption does not hold since such flow tables are implemented in Ternary Content Addressable Memory (TCAM) which is expensive and power-hungry. We consider the use of wildcard rules to compress the forwarding tables.

In this paper, we propose optimization methods to minimize energy consumption for a backbone network while respecting capacity constraints on links and rule space constraints on routers. In details, we present two exact formulations using Integer Linear Program (ILP) and introduce efficient heuristic algorithms. Based on simulations on realistic network topologies, we show that, using this smart rule space allocation, it is possible to *save almost as much power consumption as the classical EAR approach*

Key-words: Software Defined Networks, data center networks, routing tables, compression, TCAM memory, energy savings, backbone networks

* A short version of this work has been accepted for publication [12].

† This work has been partially supported by ANR program ANR-11-LABX-0031-01.

‡ University Nice Sophia Antipolis, Laboratoire I3S, UMR 7172, CNRS, INRIA, COATI

§ Department of Electronic and Electrical Engineering, University College London

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Compression de table pour le routage efficace en énergie dans les Software-Defined Networks

Résumé : Les réseaux programmable (ou Software-Defined Networks (SDN)), et en particulier OpenFlow, sont un nouveau paradigme réseau permettant l'innovation au travers de la programmation du réseau. Au cours des dernières années, plusieurs applications ont été construites grâce à SDN telles que l'équilibrage de charge de serveurs, la migration de machines virtuelles, l'ingénierie de trafic ou bien le contrôle d'accès. Dans ce papier, nous nous concentrons sur l'utilisation de SDN pour du routage vert. Puisque la charge du trafic à une faible influence sur la consommation énergétique des routeurs, le routage vert autorise la mise en veille de liens non utilisés afin d'économiser de l'énergie. SDN peut collecter la matrice de trafic pour ensuite calculer un routage satisfaisant la Qualité de Service tout en ayant une consommation énergétique minimale. Cependant, les précédents travaux sur le routage vert supposent que les tables de routage de commutateurs OpenFlow peuvent contenir une infinité des règles. En pratique, cette hypothèse ne tient pas puisque ces tables sont implémentées avec de la mémoire ternaire (TCAM) qui est cher et gourmande en énergie. Nous considérons l'utilisation de règles d'agrégation pour compresser les tables de routage.

Dans ce papier, nous proposons des méthodes d'optimisation pour minimiser la consommation d'énergie des réseaux backbone tout en respectant les contraintes de capacités des liens ainsi que les contraints des tailles de tables sur les routeurs. Nous présentons deux formulations exactes utilisant des Programmes Linéaires Entier (PLE) et introduisons des heuristiques efficaces. Grâce à des simulations sur des topologies de réseaux réels, nous montrons que, en utilisant ce placement intelligent de règles, il est possible d'*économiser autant d'énergie que dans le cas du routage vert classique.*

Mots-clés : Software Defined Networks, réseaux programmables, réseaux de centre de données, table de routage, mémoire TCAM, économie d'énergie, réseau backbone

1 Introduction

The environmental footprint of the Information and Communication Technology (ICT) sector is a growing concern. According to [26], the sector's own emissions are expected to increase to 1.43 billion tonnes carbon dioxide equivalent with 43% attributed to data centers and telecommunication networks. The reduction of CO2 emissions and the economic savings associated are thus an important issue in the scientific community.

Recent studies show that the traffic load of routers only has small influence on their energy consumption. Instead, the dominating factor is the number of active elements on routers such as ports, line cards, base chassis, etc. Therefore, in order to minimize the energy consumption, fewer network elements should be used while preserving connectivity and QoS.

Software Defined Network is a rising networking paradigm that propose a centralized management of the network, in contrast with the current decentralized networks. The approach consists in separating the control plane from the data plane. The routers and switches become simple forwarding devices while one or multiple controllers do the heavy lifting by computing paths and instructing the routers how to handle the packets in the network using the OpenFlow protocol [19]. Reshaping the traffic is thus easier on an SDN since the controllers have a total knowledge of the topology and its usage. This flexibility ease the deployment of green policies on the network. The traffic can be easily aggregated on a subset of the network with a change in the Forwarding Information Base (FIB) of the switches and the unused link shut down.

To store the forwarding rules given by the controller, the switches uses Ternary Content Address Memory (TCAM), an expensive and power hungry memory type. Moreover, the OpenFlow rules are more complex than their legacy counterpart. While the legacy rules matches on only two fields, the OpenFlow rules can match on twelve fields in its version 1.0. The size of the forwarding tables is thus a scarce resource in an SDN, the maximum number of rules in a switch can vary from 750 to a couple thousands in current hardware [1]. This limit can even be lower if we consider the use of IPv6 traffic. The capacity of the forwarding table is a non negligible constraint of the deployment of SDN.

In this paper, we use *Software Defined Networks to deploy an Energy Aware Routing (EAR)* which routes the demands on the network *respecting the capacity constraints of the links and of the forwarding tables* while minimizing the energy consumption of the network. We use *wildcard rules to reduce the size of the forwarding tables.* These wildcard rules aggregate rules with the same action on corresponding fields. We study in particular two kinds of compression of forwarding tables: *default port compression*, using a wildcard forwarding all packets to a default port, and *multi-field compression*, using additional wildcards aggregating all flows with a field having a specific value (for example, all flows going to a specific destination). We name the problem considered here, Energy Aware Routing with Compression (EARC).

Our contributions to EARC problem are:

- To our best knowledge, this is the first work that defines and formulates the optimizing rule space problem in SDN for EAR.
- We provide Integer Linear Programs to solve optimally EARC for two levels of forwarding table compression: default port compression and multi-field compression in Section 4.
- As EAR (and thus EARC) is known to be NP-hard [11], we propose heuristic algorithms for EARC that are effective for large network topologies in Section 5. The algorithm have three main modules: a compression module in charge of compressing the routing table, a routing module responsible for finding a route for each demand satisfying the capacity

constraints and an energy module deciding which network link to turn off. In particular, we propose several solutions for the compression problem.

- In Section 6, we compare the different solutions of the compression problem. We validate them on random forwading tables as well as on network tables originating from simulations on networks of the SDNLib library [21].
- Using real-life data traffic traces from SNDlib, we quantify energy savings achieved by our approaches. Moreover, we also present other QoS aspects such as routing length and link load of EAR solutions in Section 7.

2 Related Work

2.1 Classical Energy-aware Routing

Starting from the pioneering work of Gupta [14], the idea of power proportionality has gained a growing attention in networking research area [5, 6]. Since power consumption of router is independent from traffic load, people suggested putting network components to sleep in order to save energy. Although power savings is worthwhile, performance effects must be minimal, and fault tolerance must be satisfied. Several proposals have been proposed to find feasible routing solutions while satisfying QoS constraints and being minimal in power consumption. For instance, the authors in [5, 6] use mixed integer programming to optimize router power in a wide area network. Furthermore, other works on saving energy for data centers have also been presented [15, 25]. In general, these works show that up to 50% of network energy can be saved while maintaining the ability to handle traffic surges and guaranteeing QoS.

2.2 Limited Rule Space in OpenFlow Switches

To support a vast range of network applications, OpenFlow rules are more complex than forwarding rules in traditional IP routers. For instance, access-control requires matching on source - destination IP addresses, port numbers and protocol [4] whereas a load balancer may match only on source and destination IP prefixes [24]. These complicated matching can be well supported using TCAM since all rules can be read in parallel to identify the matching entries for each packet. However, as TCAM is expensive and extremely power-hungry, the on-chip TCAM size is typically limited. Several works have tackled the distribution of the forwarding policies on a network considering the table size constraints. In [18] and [17], the authors propose similar solutions in which the set of end points policies of the network is divided and then spread over the network so that every packet is affected by all the policies. However the routing policies are not taken into consideration. In both [20] and [8], routing policies are dealt with by changing the path of the flows to take advantages of the table space from all the switches of the network. These types of solution do however change the path used to route a flow and thus impact the QoS of the network. [16, 3] use compression methods to deal with limited rule space, as we do in our work. In [16] the authors introduce XPath which identifies end-to-end paths using path ID and then compresses all the rules and pre-install the necessary rules into the TCAM memory. This solution requires to contact the controller for every new flow entering the network to obtain the corresponding path ID. In [3], the authors suggest to follow the concept of longest prefix matching with priorities for compression, using the Espresso [23] heuristic. They show that their algorithm leads to 17% savings only. In this paper, we use different compression methods, which are very efficient, as we use wildcards to aggregate rules based on several fields. This leads to savings over 80% in terms of number of rules for practical network forwarding tables. The

compression solution using multi-field compression was tested with experiments on a small SDN platform for data center networks in [22]. The experiments show that this is a realistic solution and that the impact of compression on failure rate and delay is negligible. In the present work, we extend this work by proposing new solutions for the combined problem of routing, compressing and minimizing the energy consumption, in the new context of ISP networks.

2.3 Energy Savings with OpenFlow

Since, as stated, power consumption of router is largely independent from traffic load, people suggested putting network components to sleep in order to save energy. OpenFlow is a promising method to implement EAR in a network. Without setting entries manually, OpenFlow can collect traffic matrix, performs routing calculation and then installs new routing rules on routers. For instance, the authors in [15] have implemented and analyzed ElasticTree on a prototype testbed built with production OpenFlow switches. The idea is to use OpenFlow to control traffic flows so that it minimizes the number of used network elements to save energy. Similarly, the authors in [25] have set up a small testbed using OpenFlow switches to evaluate energy savings for their model. OpenFlow switches have also been mentioned in existing work as an example of the traffic engineering method to implement the EAR idea [9]. However, as we can see, the testbed setups with real OpenFlow switches are quite small. For instance, in [15], 45 virtual switches onto two 144-port 5406 chassis switches are used; or in [25], there is a testbed with 10 virtual switches on a 48-port Pronto 3240 OpenFlow-enabled switch. We argue that when deploying EAR in real network topologies, much more real OpenFlow switches should be used and they have to handle a large amount of traffic flows. In this situation, limited rule space in switches becomes a serious problem since we can not route traffic as expected. Therefore, we present in next sections a novel optimization method to overcome the rule placement problem of OpenFlow for EAR.

3 Problem definition

Energy Aware Routing with Compression (EARC) We consider a backbone network as an *directed* graph $G = (V, A)$. The nodes in V describe routers and the arcs in A represent connections or links between those routers. The links have a limited capacity. We denote by C_{uv} the capacity of a link (u, v). The nodes have a limited memory space to store rules and we note C_u the maximum number of rules can be installed at router u. We denote by D^{st} the demand of traffic flow from node s to node t such that $D^{st} \geq 0, s, t \in V, s \neq t$. The *objective is to find a feasible routing for all traffic flows, respecting the capacity and the rule space constraints and being minimal in energy consumption.* We name the problem *Energy Aware Routing with Compression (EARC).* Since power consumption of routers is mostly independent from traffic load as stated in related work, the energy consumption of the network is given by the number of active links in our model. We consider that routers have to stay powered on in backbone networks as they are the points of entry and of exit of traffic.

Compression problem A forwarding table is composed of multiple entries that match flows with corresponding action(s). In OpenFlow 1.0, the matching can be done on 12 fields from the packet header. For each field, the matching rule can use a specific value or a wildcard (noted $*$) that can accept any value. The action associated with a matching rule can be to drop the packet, modify the header, or forward to a specific port.

In the following, we consider the action to be limited to a forward to outgoing ports. We also limit the use of the wildcard to the source and destination addresses. However, our solution also

Flow	Output port
(0, 4)	Port-4
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 5)	Port-4
(1, 6)	Port-6
(2, 4)	Port-4
(2, 5)	Port-5
(2, 6)	Port-6

(a) Without Compression

Flow	Output port
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 6)	Port-6
(2, 5)	Port-5
(2, 6)	Port-6
$(*, *)$	Port-4

(b) Default port compression (only $(*, *)$ rule)

Flow	Output port
$(0, 4)$	Port-4
$(1, 5)$	Port-4
$(2, 4)$	Port-4
$(2, 5)$	Port-5
$(0, *)$	Port-5
$(*, *)$	Port-6

(c) Multi-field compression with only $(n, *)$ and $(*, *)$ rules

Flow	Output port
(1, 4)	Port-6
$(1, 5)$	Port-4
$(0, 6)$	Port-5
$(*, 4)$	Port-4
$(*, 5)$	Port-5
$(*, *)$	Port-6

(d) Multi-field compression with only $(*, n)$ and $(*, *)$ rules

Flow	Output port
(1, 5)	Port-4
(2, 6)	Port-6
$(1, *)$	Port-6
$(*, 4)$	Port-4
$(*, *)$	Port-5

(e) Optimal solution using multi-field compression

Table 1: Examples of routing tables: (a) without compression, (b) default port compression, (c) multi-field compression with compression by the source and default rule, (d) multi-field compression with compression by the destination and default rule, and (e) multi-field compression with the three aggregation rules, giving the routing table with minimum number of rules.

applies if other fields are considered such as ToS (Type of Service) field or transport protocol. We compress a table by using either the aggregation by source (i.e $(s, *, p)$), by destination (i.e $(*, t, p)$) or by the default rule (i.e $(*, *, p)$). When only the default rule is used, we talk of *default port compression*, and, when all the wildcard may be used, of *multi-field compression.*

Table 1 represents the possible versions of the same table with the use of the different wildcards. Table 1(a) represent the original table while in Table 1(b), we only use the default port rule, in Table 1(c), the aggregation by source (and default port rule) and in Table 1(d), the aggregation by destination (and default port rule) . Finally, in Table 1(e), we use all three types of wildcard rules and obtain the optimal compression table. *Since multiple entries can correspond to the same flow, rules are considered in the order of the table. A rule on top of the table has priority over a rule below.* If we exchange the priorities of the rules $(1, *, Port - 6)$ and $(*, 4, Port - 4)$, a flow between 1 and 4 is no longer forwarded through the Port-4 like in the original table.

4 Integer Linear Programming

We propose two Integer Linear Programs to solve the EARC problem. In the first one, *EARC-LP-Default*, only the default port compression is allowed, while multi-field compression is used in the second, *EARC-LP-Multi*. The first program thus is less powerful but runs faster. We were able to obtain optimal solutions for small networks using both ILPs.

The following notations are used in both formulations:

- x_{uv}: binary variable to indicate if the link (u, v) is active or not.
- $\mathcal{D}$: the set of all traffic demands to be routed.
- $D^{st} \in \mathcal{D}$: demand of traffic flow from s to t.
- C_{uv}: capacity of a link (u, v).
- C_u: maximum number of rules can be installed at router u.

4.1 EARC with default port Compression (EARC-LP-Default)

In this version of the problem, a flow can be routed following the FIB, that contain only *perfect match* rules, or via the default port. The following notations are used for the formulation of the ILP:

- f_{uv}^{st}: a flow (s, t) that is routed on the link (u, v) by a distinct rule. We call f_{uv}^{st} as normal flow.
- g_{uv}^{st}: a flow (s, t) that is routed on the link (u, v) by a default rule. g_{uv}^{st} is called default flow to distinguish from the normal flow f_{uv}^{st} .
- k_{uv}: binary variable to indicate if the default port of the router u is to go to v or not.

The linear program is:

$$\min \sum_{(u,v)\in E} x_{uv} \tag{1}$$

$$\text{s.t.} \sum_{v\in N(u)} \left(f_{vu}^{st} + g_{vu}^{st} - g_{uv}^{st} - f_{uv}^{st}\right) = \begin{cases} -1 & \text{if } u = s, \\ 1 & \text{if } u = t, \\ 0 & \text{else} \end{cases}$$
$$\forall u \in V, (s, t) \in \mathcal{D} \tag{2}$$

$$f_{uv}^{st} + g_{uv}^{st} \leq 1 \; \forall (u,v) \in E, (s,t) \in \mathcal{D} \tag{3}$$

$$\sum_{(s,t)\in\mathcal{D}} D^{st}(f_{uv}^{st} + g_{uv}^{st}) \leq C_{uv}x_{uv} \quad \forall (u,v) \in E \tag{4}$$

$$\sum_{(s,t)\in\mathcal{D}} \sum_{v\in N(u)} f_{uv}^{st} \leq C_u - 1 \quad \forall u \in V \tag{5}$$

$$\sum_{v\in N(u)} k_{uv} \leq 1 \quad \forall u \in V \tag{6}$$

$$g_{uv}^{st} \leq k_{uv} \quad \forall (u,v) \in E, (s,t) \in \mathcal{D} \tag{7}$$

$$x_{uv}, f_{uv}^{st}, g_{uv}^{st}, k_{uv} \in \{0,1\} \quad \forall (u,v) \in E, (s,t) \in \mathcal{D} \tag{8}$$

The objective function (1) minimizes the power consumption of the active links. The flow conservation constraints (2) express that the total flows entering and leaving a router are equal (except the source and the destination nodes). It is noted that a normal flow entering a router can become a default flow on outgoing link and vice versa. Constraints (3) ensure that a flow (s,t) on a link (u,v) cannot be both normal (f_{uv}^{st}) and default flow (g_{uv}^{st}) at the same time. Constraints (4) are capacity constraints. Constraints (5) denote rule capacity constraints where we reserve one rule at each router to be the default rule. Constraints (6) and (7) are used to fix only one default port for each router.

4.2 EARC with multi-field Compression (EARC-LP-Multi)

In this version, we consider that the forwarding table several wildcard rules. These rules can match any flow that comes from a source s (i.e. $(s,*,p)$) or goes to a destination t (i.e. $(*,t,p)$). The following notations are used for the formulation of the ILP:

- $\mathcal{S}$, set of all sources
- $\mathcal{T}$, set of all destinations
- F_{uv}^{st}: binary variable to indicate a flow (s,t) that is routed on the link (u,v)
- $r_u(s,t,p)$, binary variable to indicate if the rule (s,t,p) exists on the router u
- $gs_u(t,p)$, binary variable to indicate if the wildcard rule $(*,t,p)$ exists on the router u
- $gd_u(s,p)$, binary variable to indicate if the wildcard rule $(s,*,p)$ exists on the router u
- $dp_u(p)$, binary variable to indicate if p if the default port on the router u
- $order_u(r_s, c_t)$, binary variable to indicate if the wildcard rule for the row s has higher priority than the one for the column t.

$$\min \sum_{(u,v)\in E} x_{uv} \tag{9}$$

$$\text{s.t.} \sum_{v \in N(u)} \left(F^{st}_{vu} - F^{st}_{uv}\right) = \begin{cases} -1 & \text{if } u = s, \\ 1 & \text{if } u = t, \\ 0 & \text{else} \end{cases}$$
$$\forall u \in V, (s,t) \in \mathcal{D} \tag{10}$$

$$\sum_{(s,t) \in \mathcal{D}} D^{st} F^{st}_{uv} \leq C_{uv} x_{uv}$$
$$\forall (u,v) \in E \tag{11}$$
$$\sum_{v \in N(u)} dp_u(v) + \sum_{(s,t) \in \mathcal{D}} r_u(s,t,v) + \sum_{t \in \mathcal{T}} gs_u(t,v) + \sum_{s \in \mathcal{S}} gd_u(s,v) \leq C_u$$
$$\forall u \in V \tag{12}$$
$$r_u(s,t,v) + gs_u(t,v) + gd_u(s,v) + dp_u(v) \geq F^{st}_{uv}$$
$$\forall (u,v) \in A, (s,t) \in \mathcal{D} \tag{13}$$
$$\sum_{v \in N(u)} dp_u(v) \leq 1$$
$$\forall u \in V \tag{14}$$
$$\sum_{v \in N(u)} gs_u(t,v) \leq 1$$
$$\forall u \in V, t \in \mathcal{S} \tag{15}$$
$$\sum_{v \in N(u)} gd_u(s,v) \leq 1$$
$$\forall u \in V, s \in \mathcal{S} \tag{16}$$
$$order_u(c_s, r_t) = 1 - order_u(r_t, c_s)$$
$$\forall u \in V, s \in \mathcal{S}, t \in \mathcal{T} \tag{17}$$
$$r_u(s,t,v_1) + \frac{gs_u(t,v_1) + order_u(c_t, r_s)}{2} \geq gd_u(s,v_2)$$
$$\forall u \in V, (s,t) \in \mathcal{D}, v_1, v_2 \in N(u), v_1 \neq v_2 \tag{18}$$
$$r_u(s,t,v_1) + \frac{gd_u(s,v_1) + order_u(r_s, c_t)}{2} \geq gs_u(t,v_2)$$
$$\forall u \in V, (s,t) \in \mathcal{D}, v_1, v_2 \in N(u), v_1 \neq v_2 \tag{19}$$
$$1 \leq order_u(c_{s_1}, r_{t_1}) + order_u(r_{t_1}, c_{s_2})$$
$$+order_u(c_{s_2}, r_{t_2}) + order_u(r_{t_2}, c_{s_1}) \leq 3$$
$$\forall u \in V, s_1, s_2 \in \mathcal{S}, t_1, t_2 \in \mathcal{T} \wedge s_1 \neq s_2 \wedge t_1 \neq t_2 \tag{20}$$

$$x_{uv}, F^{st}_{uv}, r_u(s,t,v), gd_u(s,v), gs_u(t,v), dp_u(v), order_u(r_s, c_t) \in \{0,1\}$$
$$\forall (u,v) \in E, (s,t) \in \mathcal{D} \tag{21}$$
$$\tag{22}$$

The objective (9) is to minimize the number of active links in the network. Constraints (10) express the flow conservations on the network. The link and table capacities are respectively ensured with constraints (11) and (12). Constraints (13) to (19) establish the validity of the compressed table on each router. Constraints (13) warrant that there exists at least one rule to forward the flow. Constraint (14) ensure that there is exactly one default port on each router. Constraints (15) (resp. (16)) denote that, for every destination t (resp. source s), there is at most one rule $(*, t, p)$ (resp. $(s, *, p)$). Two rules cannot have simultaneously a higher priority than the other, Constraints (17). Constraints (18) and (19) warrant that for every rule (s, t, p) in the original table, if a wildcard $(s, *, p' \neq p)$ or $(*, t, p' \neq p)$ exists, either the original (s, t, p) exists or a corresponding wildcard rule exists with a higher priority. Last, there are no cyclic order dependencies between rules thanks to Constraints (20).

Both linear programs run for small networks. In particular, we were able to obtain optimal solutions for the Atlanta network from SNDLib, which has 15 nodes and 22 bi-directional links, see Section 7. However, the running time increases very quickly as the Energy Aware Routing problem is NP-Hard [11]. Thus, we propose efficient heuristic algorithms for larger networks in the following section.

5 Heuristic Algorithms

As the linear programs proposed in the previous section do not run for medium and large networks, we propose here efficient heuristic algorithms. The problem can be decomposed into three sub-problems:

- First, the *compression problem* consists in reducing the size of a single table by using aggregation rules: the default rule for default port compression, and, additionaly, source or destination rules for the multi-field compression.
- Second, the *routing problem* goal is to compute and assign a path in the network for each demand, while respecting the link and forwarding table capacities.
- Last, the *energy saving problem* goal is to shut down a maximum number of links while maintaining a valid routing in the network for all the flows.

The heuristic algorithm is thus composed of three different modules designed to solve these sub-problems. For the compression module, we propose multiple heuristics for the two levels of compression (default port and multi-field compression).

5.1 Compression module

We propose several solutions to solve the compression problem: First, *Comp-Default*, giving optimal solutions for the default port compression. We then provide an integer linear program, *Comp-LP*, which gives optimal solutions for the multi-field compression. However, as the problem is NP-Hard (see [10] for a proof), the program does not scale to large tables (also see Section 6 for compression time and a discussion). We thus provide two heuristic algorithms for the compression problem, *Comp-Greedy* and *Comp-Direction*.

5.1.1 Default Rule (Comp-Default)

When using only the default port compression, finding the optimal solution is simple. The algorithm finds the most occuring port p^* in the forwarding table, remove all the rules with p^*, and add the default rule $(*, *, p^*)$ at the end of the table.

5.1.2 Integer Linear Programming (Comp-LP)

We first define the following notations. We then formulate the problem as an Integer Linear Program. Note that a large number of constraints are similar to the one of EARC-LP-Multi. We yet decided to provide here the full program to avoid confusion, even if it is at the cost of some repetitions for the reader.

- $\mathcal{R}$, set of rules in the forwarding table
- $\mathcal{S}$, set of sources in the forwarding table
- $\mathcal{T}$, set of destinations in the forwarding table
- $\mathcal{P}$, set of ports of the router
- $r(s, t, p)$, binary variable to indicate if the rule (s, t, p) exists
- $gs(t, p)$, binary variable to indicate if the wildcard rule $(*, t, p)$ exists
- $gd(s, p)$, binary variable to indicate if the wildcard rule $(s, *, p)$ exists
- $dp(p)$, binary variable to indicate if p if the default port
- $order(r_s, c_t)$, binary variable to indicate if the wildcard rule for the row s has higher priority than the one for the column t.

$$\min 1 + \sum_{(s,t)\to p \in \mathcal{R}} r(s,t,p) + gs(t,p) + gd(s,p) \tag{23}$$

$$\text{s.t. } r(s,t,p) + gs(t,p) + gd(s,p) + dp(p) \geq 1$$
$$\forall (s,t,p) \in \mathcal{R} \tag{24}$$

$$\sum_{p\in\mathcal{P}} dp(p) = 1 \tag{25}$$

$$\sum_{p\in\mathcal{P}} gs(t,p) \leq 1$$
$$\forall t \in \mathcal{T} \tag{26}$$

$$\sum_{p\in\mathcal{P}} gd(s,p) \leq 1$$
$$\forall s \in \mathcal{S} \tag{27}$$

$$order(c_s, r_t) = 1 - order(r_t, c_s)$$
$$\forall s \in \mathcal{S}, t \in \mathcal{T} \tag{28}$$

$$r(s,t,p_1) + \frac{gs(t,p1) + order(c_t, r_s)}{2} \geq gd(s,p_2)$$
$$\forall (s,t) \to p_1 \in \mathcal{R}, p_2 \in \mathcal{P} \tag{29}$$

$$r(s,t,p_1) + \frac{gd(s,p1) + order(r_s, c_t)}{2} \geq gs(t,p_2)$$
$$\forall (s,t) \to p_1 \in \mathcal{R}, p_2 \in \mathcal{P} \tag{30}$$

$$1 \leq order(c_{s_1}, r_{t_1}) + order(r_{t_1}, c_{s_2})$$
$$+order(c_{s_2}, r_{t_2}) + order(r_{t_2}, c_{s_1}) \leq 3$$
$$\forall s_1, s_2 \in \mathcal{S}, t_1, t_2 \in \mathcal{T}, s_1 \neq s_2, t_1 \neq t_2 \tag{31}$$

The objective function (23) minimizes the total number of rules in the compressed table. Constraints (24) ensure that for every rule in the table, at least a corresponding rule exists in the compressed table. Constraint (25) expresses that there is exactly one default port. Constraints (26) (resp. (27)) establish that, for every destination t (resp. source s), there is at most one rule $(*, t, p)$ (resp. $(s, *, p)$). Two rules cannot have simultaneously a higher priority than the other, Constraints (28). Constraints (29) and (30) warrant that for every rule (s, t, p) in the original table, if a wildcard $(s, *, p' \neq p)$ or $(*, t, p' \neq p)$ exists, either the original (s, t, p) exists or a corresponding wildcard rule exists with a higher priority. Last, there are no cyclic order dependencies between rules thanks to Constraints (31).

5.1.3 Most savings heuristic (Comp-Greedy)

For this heuristic algorithm, we add wildcard rules for sources or destination in a greedy way, based on the highest potential compression ratio. The *potential compression ratio* of a source s (or destination t) is equal to the number of rules with the most repeated port p among all the rules with s (or t) over the total number of rules with s (or t). At each step, we compute the potential compression ratio of all sources and destinations. We then add the wildcard rule corresponding to the source or destination with the highest potential compression ratio and we remove all the rules matching the wildcard rule. Note that at each step, the compression ratios of the other sources can be affected. We thus recompute them at each step.

5.1.4 Direction Based Heuristic (Comp-Direction or Comp-Dir in short)

We present a second heuristic which is a 3-approximation of the compression problem [10].

This heuristic computes three different compressed routing tables and, then, chooses the smallest one. An example can be seen in Figure 1. We consider the routing table given in Fig. 1(a).

To compute the first compressed table, the algorithm considers all the sources one by one. For each source s, it finds the most occurring port p^*, and replace all the matching rules with $(s, *, p^*)$. The remaining rules $(s, t, p \neq p^*)$ stay unchanged and have priority over the wildcard rule. Once all the sources have been considered, we do a pass over all the wildcard rules. We aggregate them by finding the most occurring port in the wildcard rules and by setting it as default port. The default port rule has the lowest priority of all the rules. The first compressed table is given in Fig. 1(c).

The second compressed routing table is obtained with the same method, but when considering an aggregation by destinations ($(*, t, p^*)$ rules) and not by sources. The second compressed table can be seen in Fig. 1(d).

The third and last one (Fig. 1(b)) is the result of a single aggregation using the best default port.

5.2 Routing module

The *routing module takes as input a sub-network H given by the energy savings module and tries to find a feasible routing.* Its principle is to try to *spread the flows over the sub-network* as much as possible in order to avoid to overload a link or a routing table.

The module uses a *shortest-path algorithm* with an *adaptive metric in a residual graph H^R*. At the beginning of the algorithm $H^R = H$. We route the demands one by one. Consider we have already routed k demands and that we are in step k. The residual graph is H_k^R. For a demand between two nodes s and t of load d, we create a new subgraph $H_k'^R$ by first removing all arcs with capacities lower than d. We then consider routers with a full forwarding table. For such a router u, we distinguish two cases. If its forwarding table does not contain any wildcard rule which could route the demand (s, t), we remove the router u from the residual graph. On the contrary, if such a wildcard rule exists (it should be of the form $(s, *, p)$, $(*, t, p)$ or $(*, *, p)$, where p represents also the output port of u towards v. Note that several wildcard rules may be present in the forwarding table of u. In this case, we consider the first one in the order of priority, as it will be the one routing the demand (s, t)), we keep the router u in the residual graph, but we remove all outgoing links corresponding to a port different than p.

We then compute a route by finding a shortest path between s and t in $H_k'^R$ with set of weights defined below.

The weight w_{uv} of a link depends (1) on the total flow using the link corresponding to demands previously routed, and (2) on the table's usage of router u also corresponding to demands previously routed. We note w_{uv}^c the weight corresponding to the link capacity and w_{uv}^r the weight corresponding to the rule capacity. They are defined as follows:

$$w_{uv}^c = \frac{\mathcal{F}_{uv} + d}{C_{uv}}$$

where C_{uv} is the capacity of the link (u, v) and $\mathcal{F}_{uv}$ the total flow on (u, v). The more the link is used, the heavier the weight is. This favors the use of links with lower load, leading to

load-balancing. And

$$w^r_{uv} = \begin{cases} \frac{|R_u|}{S_u} & \text{if } \nexists \text{ wildcard rule for } (s,t,v) \\ 0 & \text{otherwise} \end{cases}$$

where S_u is the maximum table size of router u and R_u is the set of rules for router u. The weight is proportional to the usage of the table.

Finally, the weight w_{uv} of a link (u,v) is given by:

$$w_{uv} = 1 + \alpha w^c_{uv} + \beta w^r_{uv} \tag{32}$$

If a path is found for a demand, we build the residual graph H^R_{k+1} for the next step of the algorithm. For each arc (u,v) of the path, we add the rule (s,t,v) to the router u if no corresponding wildcard rule exists. If the table is full, one of the compression methods previously described is used to reduce the size of the table[1]. We also reduce the capacity of the arc by d from the one in H^R_k. If no path is found, the routing module returns that no feasible routing was found.

Setting the parameters of routing module. The metric to find a path for each demand (Equation 32) combines link usage and link capacity with table capacity and table usage. The importance of links is given by the parameter α and the one of tables by β. If α is larger than β, we give more weight to links. We compared several choices of metrics in the appendix for different networks. The best metrics is 3:1. We thus choose this metric for the remaining of the paper.

5.3 Energy savings Module

The energy savings module uses a greedy approach to select the links to switch off. It tries to remove in priority links that are less loaded and to accomodate their traffic on other links in order to reduce the total number of active links.

The algorithm is simple. We start with the full network. We launch the routing module to try to find a feasible routing for all the demands. If such a routing exists, we try to remove the edge with the lowest load. We then re-launch the routing module on the network without the considered edge. If a feasible routing is found, we continue and try to switch off another edge. If no feasible routing is found, we put back the edge, and we try to remove the edge with the second lowest load. An edge, which was selected and could not be removed, is not considered anymore in the following of the algorithm. The algorithm stops when all edges have been selected once.

6 Compression of forwarding tables

We test here the compression module. The goal is to evaluate typical compression ratios, compression times and to compare the different solutions proposed in Section 5.1: the solution using default port compression (Comp-Default) and the three solutions using the multi-field compression: the optimal one from the Linear Program (Comp-LP), when it is possible to compute it, the Direction Based Heuristic (referenced as Comp-Direction or Comp-Dir in short), and last, the Most savings heuristics (Comp-Greedy). As instances, we consider random routing tables as well as network routing tables coming from simulations on SNDlib instances [21].

[1]Note that, if, at some point, the compression method cannot further compress the table, the algorithm should remember this fact to avoid relaunching a useless compression each time and increasing the execution time.

6.1 Random tables

In this section, we focus on the compression of random tables. The following parameters are used to generate the random tables studied:

- the number of sources and destinations n
- the number of ports of the switch p
- the density of the corresponding matrix $0 \leq d \leq 1$

For a pair source-destination, there is an entry in the table with probability d, and in this case, the exiting port is chosen uniformly at random among the p ports.

We show the average compression ratio of the solutions proposed in Section 5.1 as a function of the parameters used to build the random matrices. We vary the number of ports in the experiments of Figure 1, the number of network nodes (corresponding to the number of sources and destinations) in Figure 2, and the table density in Figure 3. Each point represents the average of the results for 10 random forwarding tables for the comparison with the LP and 20 for the heuristics.

Gap from optimal for small tables. For small routing tables, we are able to compute the optimal compressed tables using the linear program (see Figure 1(a), Figure 2(a) and Figure 3). As an example, in Figure 1(a), we compare Comp-LP and the other three solutions on a set of random tables with $n = 15$ sources/destinations, a density of 0.5 with a number of ports between 2 and 9. Without surprise, the ILP compresses better than the other 3 solutions with 68% ratio at only two ports to 32% with nine ports. The two heuristics seems to present the same compression with a ratio of 59% at two ports and 23% at nine ports. Finally, the only use of the default port yields to the worst compression as it compresses 53% of the rules with 2 ports and only 15% at nine ports. Similarly, the difference of compression ratio in Figure 3 is between 4 and 10% when comparing the optimal solution with the Comp-Greedy and Comp-Direction heuristics. Default port is the less efficient solution with a compression ratio around 23%, when the compression ratio of Comp-Greedy and Comp-Direction heuristics is around 30%. In Figure 2(a), we vary the number of network nodes between 5 and 11. The global comparison between solutions is similar, except that, when there is a small number of network nodes, Comp-Greedy does not behave well and provides worse results than Comp-LP. The explanation is that, for small tables, Comp-Greedy adds source and destination aggregation rules that are not necessary, as a default rule works well. Because of the order between source and destination rules, most of these rules cannot be aggregated when we add the default rule, leading to an inefficiency. The problem disappears for larger numbers of network nodes (larger than 10), and thus would not appear for ISP networks which have more network nodes.

Comparison between heuristics for larger tables. However, the ILP does not scale well for larger tables. In Figure 1(b), we only compare the two heuristics and Comp-Default on tables with $n = 450$ sources/destinations and a density of 0.5. First, we notice that the two heuristics Comp-Greedy and Comp-Direction obtain the best results. However, the Comp-Default solution is not far behind with a ratio between 49% and 11% for the random tables. We will see later that the difference is significantly higher for real network tables. Comp-Greedy behaves better than Comp-Direction, with a compression ratio between 55% and 16% to be compared to a compression ratio between 52% and 14% for Comp-Direction.

Impact of the parameters. The compression ratio is very sensitive to the number of ports, see Figure 1. The compression ratio varies from 55% to 18% when the number of ports varies from 2 to 9 for a random matrice with around 100,000 rules. Similar results are observed for

small table with variations from 70% to 35%. *We observe higher compression ratio for smaller number of ports.* This is expected as, for example, the impact of setting a default port is higher when the number of port is lower. For two ports, using a default port saves at least 50% of the rules.

Conversely, the *density and the size (number of network nodes) of the forwarding tables do not have an important impact on the compression ratio.* For the experiments in Figures 2, the compression ratio varies of only few percents when the number of network nodes increases from 5 to 11, and then from 50 to 1000; and similarly, when the density goes from 0.1 to 1, even if it represents a 10-fold increase of the number of rules in the table (Figure 3). However, density and size of the forwarding tables have an impact on the compression time as discussed below.

Compression time. We study the time to compress forwarding tables. This time depends mostly of the number of entries in the forwarding table, as presented in Figure 4. The compression time using linear programming (Comp-LP) is a lot higher than the one using heuristic algorithm: around 1000 s for only 125 rules, when it takes a lot less than 1 ms for the heuristics. We thus had to present the results for Comp-LP independently in Figure 4(a) with a different log-scale ($[0, 10^7]$), compared to ($[0, 10^4]$) for Figure 4(b). We observe that the compressing time of Comp-LP increases exponentially with the number of rules. It reaches the limit of one hour we had set for tables with a little bit more than 150 rules. Note that a network with 10 nodes cannot have more than 90 entries in a routing table (in the extreme case of one central node seeing all the possible flows). Thus, we know that we can use Comp-LP for networks with a number of nodes reaching 10, and surely a little bit more as all traffic usually is not routed through a unique node. In fact, we show in Section 7, that LP runs on the SNDlib Atlanta network with 15 nodes, but that it is not usable for larger networks.

On the contrary, the *compression time of the heuristic algorithms is very low* and does not increase exponentially, but linearly in the number of rules. A large network with 100 nodes cannot have more than 10,000 entries in a routing table. A forwarding table of this size is compressed in less than 10 ms (around 10 ms for Comp-Greedy, 1 ms for Comp-Direction, and less than 1 ms for Comp-Default). It is even possible to compress a routing table of size 1M rules (for a network of more than a thousand nodes) in a little bit more than 1 s for Comp-Greedy and less than 10 ms for Comp-Direction and Comp-Default. The heuristic algorithms for compression can thus be used for very large networks and have a very low execution time.

6.2 Network tables

We now compare the solutions on tables from routing on backbone networks using the routing and compression module presented in Section 5.1 and 5.2. We use four of the SNDlib instances shown in Figure 5:

- atlanta network with 15 nodes and 44 directed links,
- germany50 network with 50 nodes and 176 directed links,
- zib54 network with 54 nodes and 216 directed links, and
- ta2 network with 81 nodes and 162 directed links.

For each network, we compute a routing of all demands without considering a limit on the number of rules. We then extract the forwarding tables for all routers. We then compress each of them with the different compression solutions. Since the ILP does not scale, we only compare it with the other solutions on the atlanta network, see Figure 6(a). On the other three

Compression kind	Name	Short name in figures	Routing	Energy	Compression algo
default port	EARC-LP-Default		LP (Section 4.1)		
multi-field	EARC-LP-Multi		LP (Section 4.2)		
default port	EARC-H-Default	EARC-Default	Heur		Opt. Comp-Default (Section 5.1.1)
multi-field	EARC-H-LP		Heur		LP Comp-LP (Section 5.1.2)
multi-field	EARC-H-Greedy	EARC-Greedy	Heur		Heur Comp-Greedy (Section 5.1.3)
multi-field	EARC-H-Direction	EARC-Dir	Heur		Heur Comp-Direction (Section 5.1.4)
none	EAR		yes	yes	none (but no limit on the number of rules)
none	EAR-with-limit		yes	yes	none (with limit on the number of rules)
none	Classic Routing	CR	yes	no	none (but no limit on the number of rules)

Table 2: Names of the solutions to solve the EARC problem (and of the EAR problem without compression for comparison).

networks, we compare the EARC-H-Direction, EARC-H-Greedy and EARC-H-Default solutions, see respectively Figures 6(b),(c)(d) for germany50, ta2, and zib54 networks.

Compression Rates. The first global observation is that the solutions achieve *higher compression rates for network tables than for random tables*, with median values *around 80% for all networks*. This is good news as it shows the efficiency of the algorithms for practical cases. The explanation of this phenomenon is that real network tables have a larger number of repeating ports traffic originating from a source or going to a destination, than random matrices.

We remark that some tables show a compression ratio near 100% for all solutions for zib54 and ta2. These tables corresponds to the two routers with only one outgoing port (the two routers in black in Figure 5). Thus, only the default port can be used to route all the demands.

Comparison of the solutions. In the atlanta network, we see that the difference of efficiency between the heuristics, Comp-Direction and Comp-Greedy, and the linear program for compression, Comp-LP, is smaller than in the case of random tables. The compression rate of Comp-Direction is almost the same as the one of Comp-LP, with a median ratio of 81%. This is also good news: *real network tables are easier to compress than random tables.* We thus can suppose than the results of the heurictics on larger networks should be good. And, in fact, we obtain very high compression rates: the median is 83% for germany, 86% for ta2 and zib54.

Last, we observe that the *difference between the two levels of compression is more significative for real network tables than for random tables.* The median ratios of the Comp-Default solution is about 30% lower than the one from the Comp-Greedy heuristics. This shows the importance of considering multi-field compression.

The two heuristics using multi-field compression, *EARC-H-Direction and EARC-H-Greedy, show similar results on all networks.* While the Comp-Greedy heuristic provides better compression ratios on random tables, *the advantage for real network tables is for the Comp-Direction heuristic*: the median ratio is 4% higher for germany50, ta2, and zib54, and 8% for atlanta. We use both heuristics in the simulations of next section in which we obtain results for the EARC problem on practical network instances.

7 Energy savings

In this section, we study the energy saved over multiple periods of time and the four following networks: atlanta, germany50, ta2 and zib54. We compare the results obtained for the different solutions proposed to solve the EARC problem, the EAR problem without compression and classical routing (CR) without energy. The different solutions are summarized in Table 2. Unless specified, the limit of the forwarding table is 750 rules. We considered a typical daily pattern of

Table 3: Energy savings (in %) and computation times (in ms) for the ILPs and the heuristics on the atlanta network

Rule capacity	EARC-LP-Default		EARC-LP-Multi		EARC-H-Direction		EARC-H-LP	
	savings	time	savings	time	savings	time	savings	time
100	52.27	641 940	-	-	40.91	~ 14	40.91	3381
750	52.27	33 830	52.27	1 368 090	40.91	~ 14	40.91	3311
2000	52.27	23 640	52.27	1 748 060	40.91	~ 14	40.91	3300

traffic as shown in Figure 7. Data come from a typical France Telecom link. For each network considered, we rescale the traffic based on the traffic matrice provided by SNDib. We then divide the day into five periods, with different levels of traffic as shown in Figure 7. D1 represents the off peak hours with the least amount of traffic on the network and D5 the peak hours. We choose a small number of periods as network operators prefer to carry out as few as possible changes of configurations of their network equipements to minimize the chance of introducing errors or producing routing instability. Moreover, most of the energy savings can be achieved with a very small number of configurations, see for example [2]. Energy savings is computed as the number of links to sleep divided by the total number of links of the network ($|E|$).

The need for more space In Figure 8, we show the number of overloaded routers (with more than 750 installed rules) when applying the heuristic proposed in [11] for Energy Aware Routing. This EAR heuristic does not take into account the table size constraint. As a result, we see that for almost every traffic patterns (except for D5 on germany50), an EAR needs more than 750 rules to be deployed. In germany50, up to 10% of the devices are overloaded. For zib54, this number goes up to 11% and 16% for ta2. This confirms that in order to be able to deploy energy policies on a SDN, the table size problem needs to be resolved.

Optimal vs. Heuristic solution We compare for a small network, atlanta (15 links and 44 links), the solutions using linear programming and heuristic algorithms. We considered solutions for different rule capacities on routers : 100, 750 and 2000 rules.

Both linear programs, EARC-LP-Default LP-Default and LP-Multi, proposed in Section 4 can be run on the atlanta network (but not on larger networks such as germany50, zib54 and ta2). As expected, LP-Multi, which solves the problem using more complex wildcards, has a longer execution time as it has more variables: between 20 and 30 min for 750 and 1000 rule capacities, to be compared with 1 min23 s and 23 s for EARC-LP-Default. The running time is too long for a rule capacity of 100 because the problem is more constrained. LP-default runs in this case, but takes more than 6 minutes. Both LPs (when running) find the same optimal solution, with a savings of 52.27%. This is due to the fact that Atlanta is a small network with nodes of small degrees. The need for compression is not high and both levels of compression achieve the same results.

We ran two heuristic algorithms with two different compression modules proposed in Section 5.1, EARC-H-Direction with the Comp-Direction heuristic and EARC-H-LP, which solves optimally the compression problem each time a table has to be compressed when flows are routed. Both heuristic provide solutions of the same values, 40.91% of energy savings. However, the computation times are a lot higher for EARC-H-LP: more than 30 s compared to 7 ms for the heuristic. The computation time will prevent us from using EARC-H-LP on larger networks. As a matter of fact, we recall the compression time of tables using the LP increases exponentially,

see Figure 4. However, we observe that both solutions provide the same level of energy savings for the three rule capacities. The EARC-H-Direction heuristic is very efficient and we use it to get solutions for larger networks.

Energy savings during the day In Figure 9, we compare the multiple solutions proposed for the compression module. We also check the possibility of an SDN routing without compression (corresponding to a simple *EAR*). The ILP is not considered in the comparison as the networks are too big to be optimally resolved in an acceptable time.

Importance of compression. First, we see that, as the networks grow in size, not all heuristics give a valid SDN routing. No compression is needed to find a valid SDN routing on germany50. However, *it is impossible to find a routing satisfying the capacity constraint for zib54 and ta2 without using a compression algorithm.* Moreover, *multi-field compression should be used to find a valid routing for ta2.* Indeed, it is impossible to find a valid routing for ta2 while using only default port compression.

Results of the heuristics. On germany50, all heuristics give similar results between 52% for the peak hours and up to 65% during the night. They are all small within a margin of about 2% from one another. The EARC-H-Greedy and EARC-H-Direction heuristics show the best results and no compression gives the worst ones in all periods.

For the zib54 network, the difference between the heuristics is a little bit more visible. Between 46% and 56% is saved during the day. Once again, either the EARC-H-Greedy or EARC-H-Direction heuristics gives the best results depending of the periods. The only exception is during the D2 periods, where the EARC-H-Default compression shut about 1% more links than the other two heuristics.

Finally, in the ta2 network, the EARC-H-Greedy heuristic saves a little bit more energy than the EARC-H-Direction one as the former saves almost 2% more than the latter.

The amount of saved energy by the heuristics for each network is different. The explanation is that the order in which each link is extinguished depends on its charge. A small change in the routing thus can affect the total energy saved.

EAR vs. EARC. We compare the results of the proposed solutions with the one of the classic EAR approach in which no limit on the number of rules is considered. We show that, *by using an efficient way to route demands and compress forwarding tables, it is possible to save almost as much power consumption as the EAR approach* (curve named *No Limit* in Figure 9). Indeed, we see that for the zib54 network, we succeeded to save the same amount of energy when using the best of all solutions. The solution EARC-H-Direction alone is very close to the EAR one. Only half a percent of energy is lost for some periods of time. On germany50, the results of the heuristics are amost as good. For some periods of time, no solutions can do as well as EAR, but the difference again is only of half a percent. In general, the results of EARC-H-Greedy are withing 1% of the one of EAR. For the network ta2, the difference between EAR and our solutions is higher, but stays with 2%.

Path lengths As we shutdown links, we remove some shortest paths in the network, and thus raise the minimum delay between nodes. To study this effect, we look at the length of the paths in our EAR solutions and compare it to a routing obtained not using the energy saving module. For these comparisons, we use the Direction heuristic.

In Figure 10, we show the distribution of the stretch ratio of the path used in EARC-H-Direction compared to a classic routing (CR). The first observation is that the behavior is similar

for the three topologies: the median stretch is about 2 in the off peak hour period (corresponding to the demand D1) and decreases to about 1.3 in the peak hours (demand D5). The explanation is that, as expected, in the off peak hours, a large number of links can be switched off, and the paths are the longest. For larger demands, more links are on, and the stretch decreases.

Note that the median value is not very high. However, the third quartile value of the off peak hours is quite high: 7, 6 and 5.25 for germany50, zib54, and ta2, respectively. These values are mostly due to paths of small lengths stretched all the way trough the network to attain their destination (corresponding for example to nodes linked by a switched-off edge). Nevertheless, we show below that these somehow large values of stretch do not cause a problem of too large delays on the networks.

Delays In Figures 11, 12 and 13, we show the delay of the paths in the three networks, for both the classical routing and an EARC solution (EARC-H-Direction). We consider an optical network in which the delay is proportional to the distance [7], and we used the distances given by the geographical coordinates in SNDlib for the germany50 network. We got an average value of 1.8 ms per link. Since the coordinates are not given for the other two topologies, we used the same average value for zib54 and ta2.
The delays for the classical routing are similar for the three networks with a median of 8 ms and a maximum of 15 ms during all periods. For the EARC solution, the values are much higher. Larger delays are shown during the off peak hours as expected. The germany50 network shows the largest delays among the three topologies. The explanation is that more energy can be saved for this network. Its median delay is between 11 ms and 16 ms, and the maximum delay is below 50 ms. The delay on the two larger networks is slightly less impacted as fewer links can be turned off. The maximum delay observed on zib54 and ta2 is about 40 ms and the medians fluctuate between 14 ms and 9 ms for zib54 and 14 ms and 10 ms for ta2.

Note, that the *maximum delay observed is always below* 50 ms. This is an important fact, as this value is often chosen by Service Level Agreements (SLAs) as the maximum allowed delay for a route in a network [13]. Thus, even if new routes computed by our algorithms may experience sometimes a high value of stretch, this will not be a problem for network operators.

Link load When we turn off links, we aggregate the flows on the remaining links. The load of them is thus increased. In Figure 14, we *compare the link load of all network links (switched off and switched on) for energy aware routing and for classical routing.* In Figure 15, cumulative distributions are given considering only the switched on links. Results are provided only for off peak traffic (D1) and rush hour traffic (D5) in the first figure for clarity reason, while all five demand matrics are considered in the other.

The first observation is the percentage of links with a null load (switched off links), e.g. for Germany 62% with the demand D1 and 54% with D5. The load on the remaining links is highly increased: for Germany againg, we see that no link has a load higher than 15% for the CR for the D1 (higher than 50 % for D5) , when 45% of the links have a load higher than this value for EARC (40% for D5). Similarly for zib54 and ta2, more than 80% of the links have a load smaller than 10% for D1 for CR and of 30% for D5, when more than 50% of the switched on links have a load higher than 50% for EARC.

Note that, in the germany50 network, there is a very notable difference between the D1 and D5 period. In the off peak hours, only 30% of the switched-on links are above 75% compared to 52% in the peak hours. In the other two networks, the difference between periods as the difference as the range of energy savings are smaller. For zib54, in the off peak hours, the maximum link utilization is 86% and the minimum is 2% while in the peak hours, 19% of the link are above

this usage and the minimum is 6%.

8 Conclusion

To our best knowledge, this is the first work considering rule space constraints of OpenFlow switch in energy-aware routing (EAR). We argue that, in addition to capacity constraint, the rule space is also important as it can change the routing solution and affects QoS. We proposed solutions using forwarding table compression, defining the problem of energy-aware routing with compression (EARC) for SDN networks. We succeed in modelling the problem using Integer Linear Programs, even for complex compression for which a flow may be routed according to two packet hearder fields. We also provide efficient heuristic algoritms for large networks.

Based on simulations with real traffic traces, we show that, using wildcard rules, our smart rule allocation can *achieve high energy efficiency for a backbone network while respecting both the capacity and the rule space constraints: thanks to forwarding table compression, the energy savings are almost as high as in the case of classic EAR without a limit on the number of forwarding rules.* We also evaluate the impact of the proposed solutions on path delay. We show that, if the delay is inevitably increased, the *maximum delay* always stays below typical values given by Service Level Agreements.

9 References

References

[1] Hp 2920 switch series.

[2] J. Araujo, F. Giroire, J. Moulierac, Y. Liu, and M. Remigiusz. Energy efficient content distribution. *Computer Journal*, pages 1–18, 2016.

[3] W. Braun and M. Menth. Wildcard compression of inter-domain routing tables for openflow-based software-defined networking. In *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, pages 25–30, Sept 2014.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker. Rethinking enterprise network control. *IEEE/ACM Trans. Netw.*, 17(4):1270–1283, Aug. 2009.

[5] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[6] L. Chiaraviglio, M. Mellia, and F. Neri. Minimizing isp network energy cost: formulation and solutions. *IEEE/ACM Transactions on Networking (TON)*, 20(2):463–476, 2012.

[7] B.-Y. Choi, S. Moon, Z.-L. Zhang, K. Papagiannaki, and C. Diot. Analysis of point-to-point packet delay in an operational network. *Computer networks*, 51(13):3812–3827, 2007.

[8] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz. On the effect of forwarding table size on sdn network utilization. In *INFOCOM*, pages 1734–1742. IEEE, 2014.

[9] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high-performance networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):254–265, Aug. 2011.

[10] F. Giroire, F. Havet, and J. Moulierac. Compressing two-dimensional routing tables with order. In *INOC*, pages 1–8, 2015.

[11] F. Giroire, D. Mazauric, and J. Moulierac. *Energy Efficient Routing by Switching-Off Network Interfaces*, chapter 10 - Energy-Aware Systems and Networking for Sustainable Initiatives, pages 207–236. IGI Global, June 2012.

[12] F. Giroire, J. Moulierac, and K. Phan. Optimizing rule placement in software-defined networks for energy-aware routing. In *IEEE Global Communications Conference (GLOBECOM)*, Austin, United States, December 2014.

[13] F. Giroire, A. Nucci, N. Taft, and C. Diot. Increasing the robustness of ip backbones in the absence of optical level protection. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 1–11. IEEE, 2003.

[14] M. Gupta and S. Singh. Greening of the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26. ACM, 2003.

[15] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *NSDI*, volume 10, pages 249–264, 2010.

[16] S. Hu, K. Chen, H. Wu, W. Bai, C. Lan, H. Wang, H. Zhao, and C. Guo. Explicit path control in commodity data centers: Design and applications. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, pages 15–28, Berkeley, CA, USA, 2015. USENIX Association.

[17] N. Kang, Z. Liu, J. Rexford, and D. Walker. Optimizing the "one big switch" abstraction in software-defined networks. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 13–24, New York, NY, USA, 2013. ACM.

[18] Y. Kanizo, D. Hay, and I. Keslassy. Palette: Distributing tables in software-defined networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 545–549. IEEE, 2013.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[20] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti. OFFICER: A general Optimization Framework for OpenFlow Rule Allocation and Endpoint Policy Enforcement. In *INFOCOM*, pages 478–486. IEEE, Apr. 2015.

[21] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. *Networks*, 55(3):276–286, 2010.

[22] M. Rifai, N. Huin, C. Caillouet, F. Giroire, D. Lopez-Pacheco, J. Moulierac, and G. Urvoy-Keller. Too many sdn rules? compress them with minnie. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, Dec 2014.

[23] M. Theobald, S. M. Nowick, and T. Wu. Espresso-hf: A heuristic hazard-free minimizer for two-level logic. In *Proceedings of the 33rd Annual Design Automation Conference*, DAC '96, pages 71–76, New York, NY, USA, 1996. ACM.

[24] R. Wang, D. Butnariu, and J. Rexford. OpenFlow-based Server Load Balancing Gone Wild. In *Hot-ICE*, pages 12–12, 2011.

[25] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao. Carpo: Correlation-aware power optimization in data center networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1125–1133, March 2012.

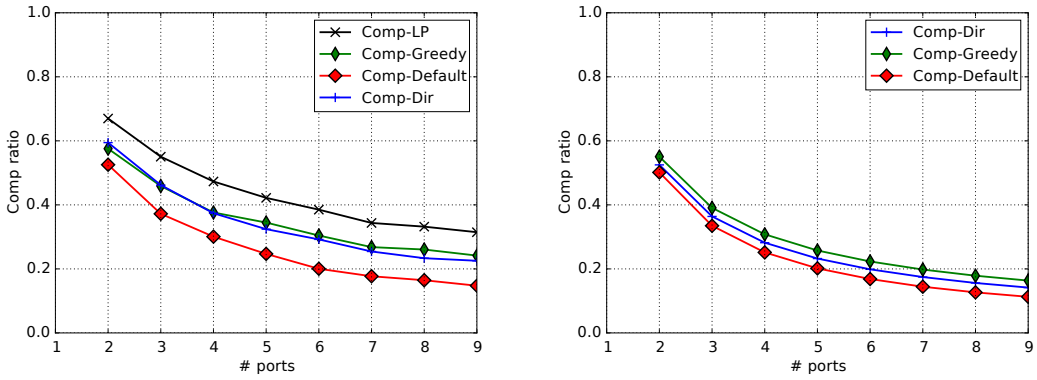
[26] M. Webb et al. Smart 2020: Enabling the low carbon economy in the information age. *The Climate Group. London*, 1(1):1–1, 2008.

A Setting the routing module metric

Before presenting the results for energy efficiency, we choose the parameters for the routing module. In Section 5.2, we proposed a custom metric to find a path for each demand (see Equation 32). In this metric, we combine link usage and link capacity with table capacity and table usage. The importance of links is given by the parameter α and the one of tables by β. If α is larger than β, we give more weight to links. In Figure 16, we compare the effect of giving a higher priority to one or the other by changing the weight α or β. In particular, we provide results for values of $\alpha : \beta$ 1:1, 3:1, and 1:3. We tested other values which are not presented here for clarity of the plots. We also compare the metric with a simple metric, called *dumb*, where all links have a weight of one.

On zib54, the use of *dumb* and *metric 1:1* allow to shutdown between 40% and 50% of the links, while the use of metric 3:1 and 1:3 allow to shutdown between 48% and 56% of the network. The same behavior can be observed on the ta2 network where between 48% and 56% of the network is shutdown with the *metric 1:1*, 52% and 56% for the dumb metric, 54% and 61% for the metric 3:1 and 56% and 60% for the metrics 1:3. We also observe that during the off peak hours, the metrics 3:1 gives better results than the metrics 1:3 while it is the other way around for the peak hours. For germany50, the difference between metrics is smaller, but the metric 3:1 is almost always the best one.

To summarize, for the three networks, the best metrics are 3:1 and 1:3. *We thus choose one of the two,* metric 3:1*, as the default metric in the remaining of the paper.*



(a) ~ 112 rules ($n = 15, d = 0.5$) (b) $\sim 101\,250$ rules ($n = 450, d = 0.5$)

Figure 1: Compression ratio as a function of the number of ports for the four compression methods.

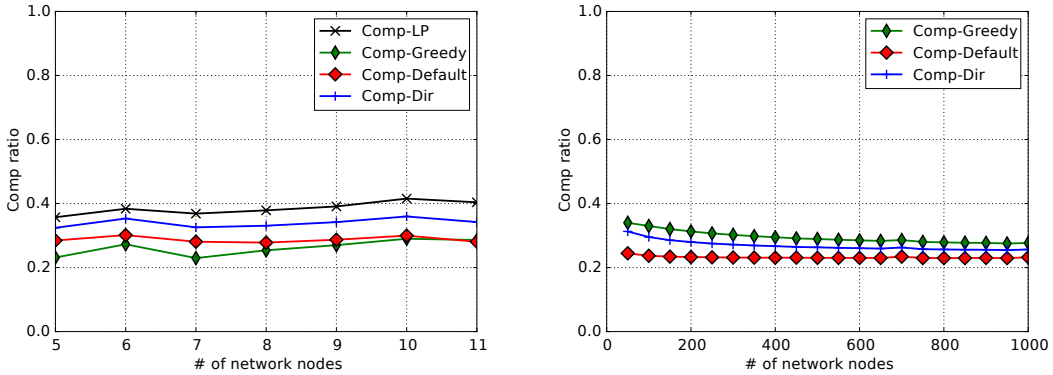


Figure 2: Compression ratio as a function of the number of network nodes (that is the number of sources and destinations) for the four compression methods.

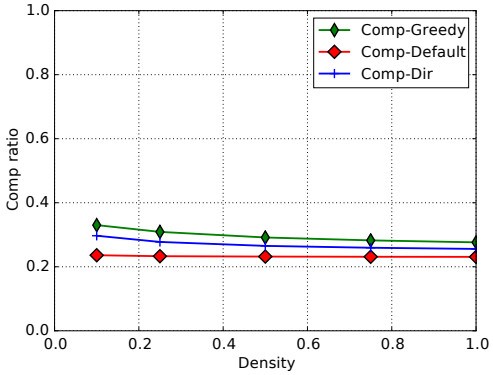
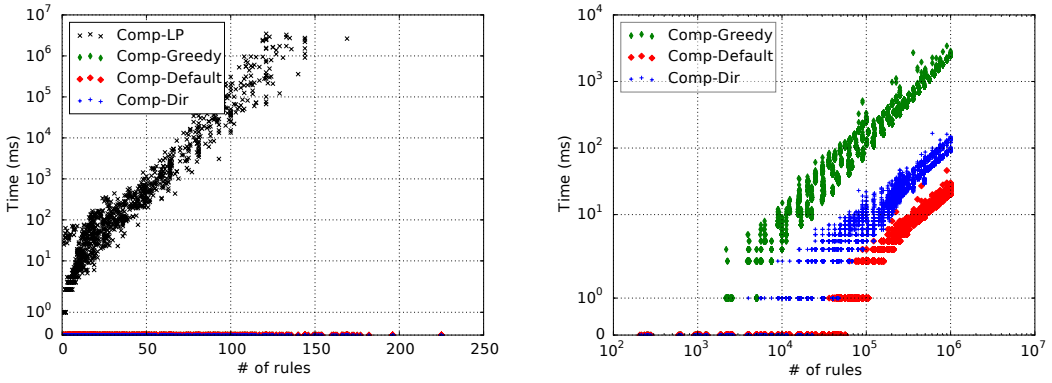


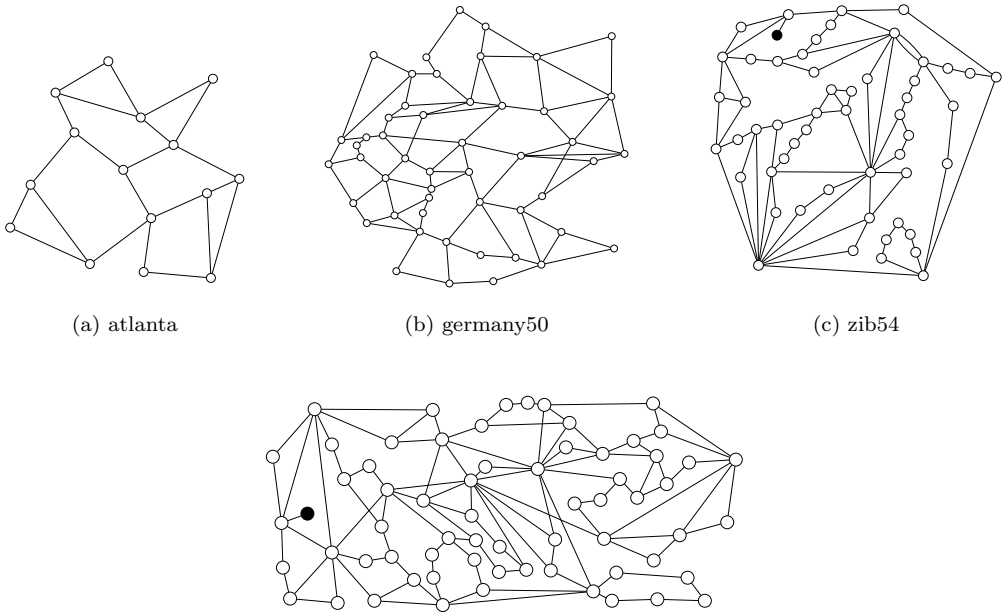
Figure 3: Compression ratio as a function of the forwarding tables density for the four compression methods.



(a) Linear Program

(b) Heuristic Algorithms

Figure 4: Compression times of forwarding tables as a function of the number of rules in the tables for four methods of compression (two different scales for Time).



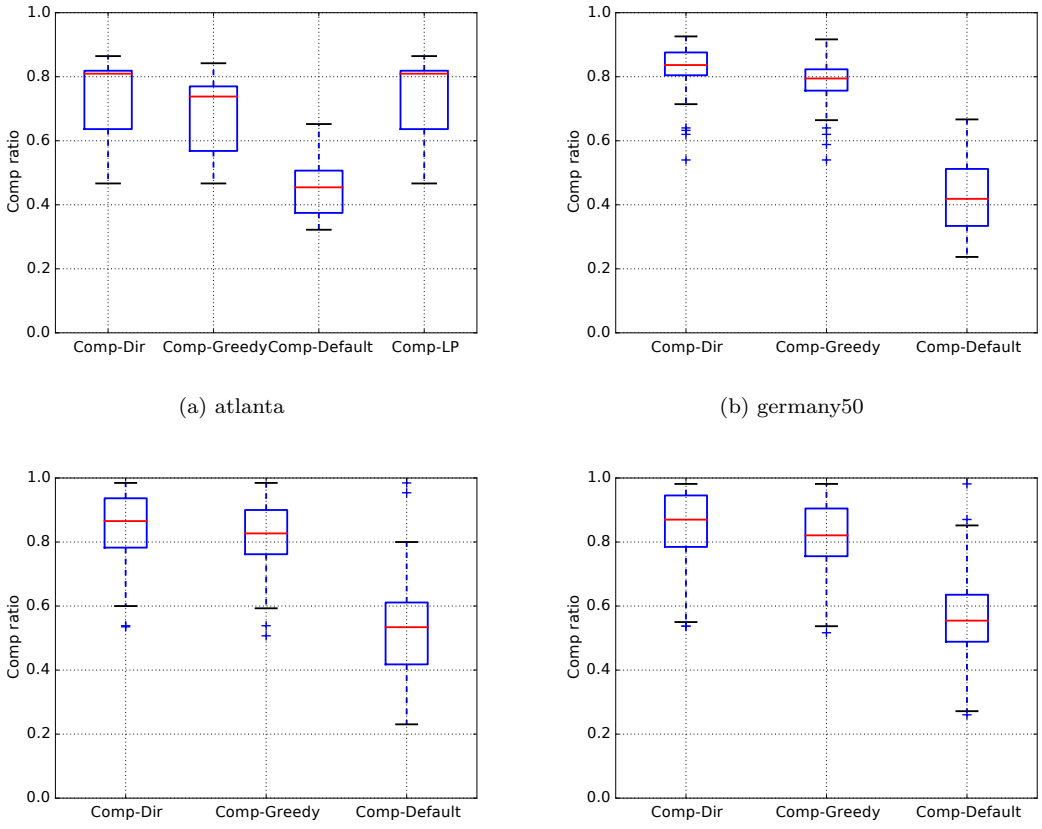
(a) atlanta

(b) germany50

(c) zib54

(d) ta2

Figure 5: The four SNDlib topologies used. Each edge corresponds to two directional links.



(a) atlanta

(b) germany50

(c) ta2

(d) zib54

Figure 6: Compression ratio of the three different heuristics on 4 different SNDlib topologies.

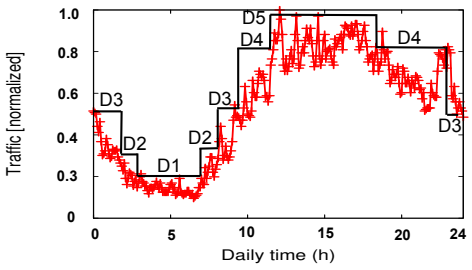
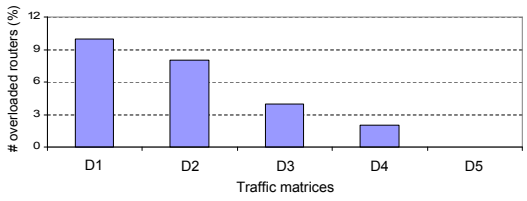
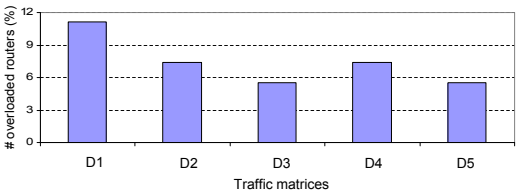


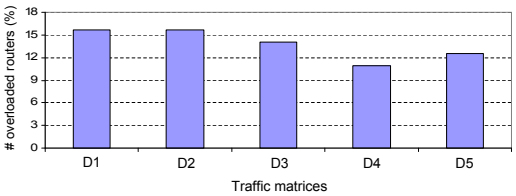
Figure 7: Daily traffic in multi-period



(a) Germany50 network

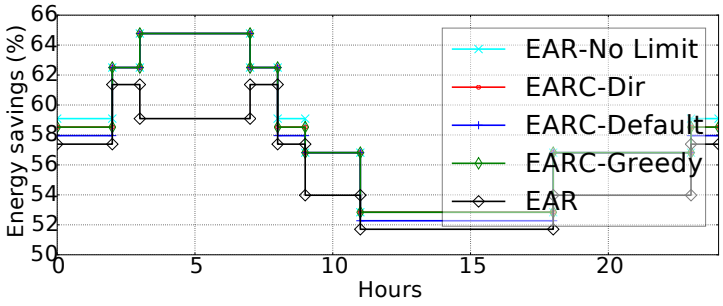


(b) Zib54 network

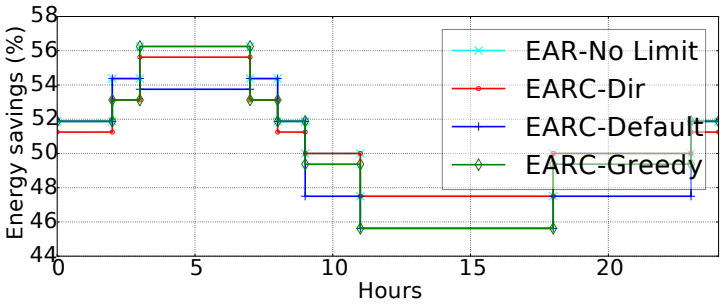


(c) Ta2 network

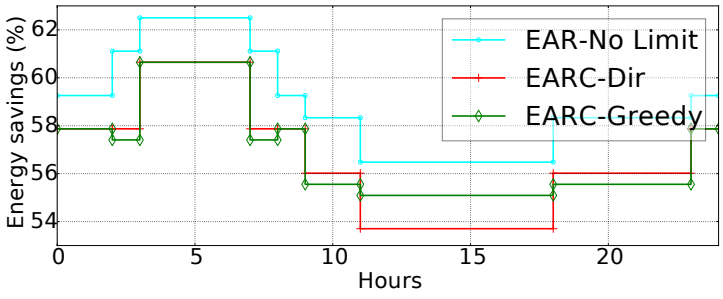
Figure 8: Number of overloaded routers in three networks with unlimited rule-space algorithm



(a) germany50

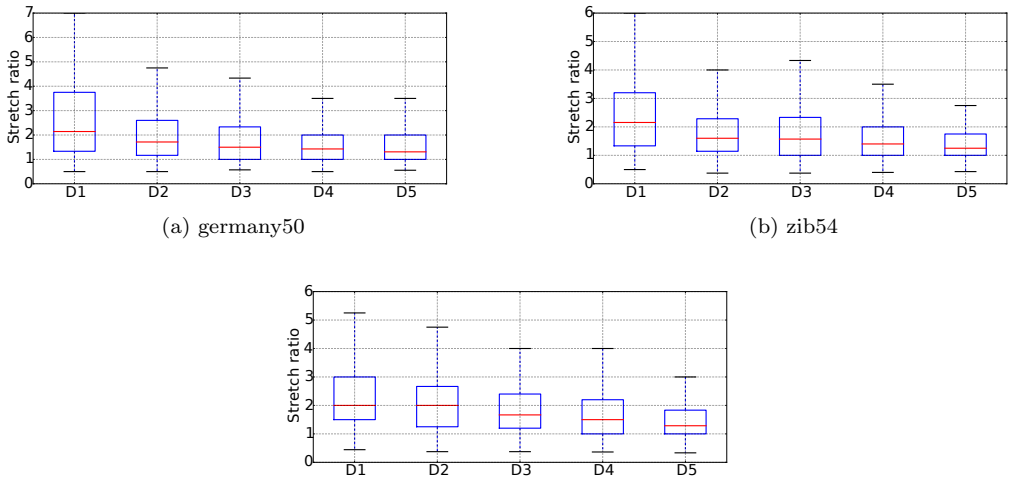


(b) zib54



(c) ta2

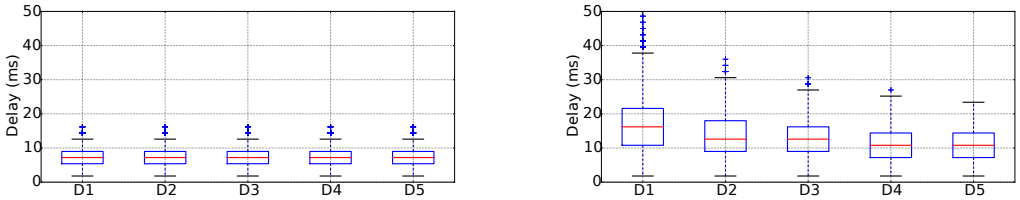
Figure 9: Energy savings of the different heuristics during the day with a limit of 750 rules.



(a) germany50 (b) zib54

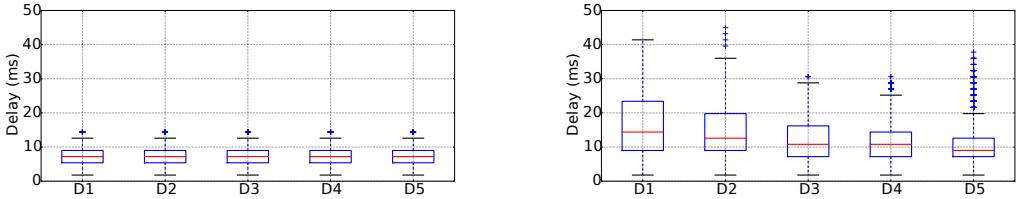
(c) ta2

Figure 10: Stretch ratio of the paths given by a EARC solution (EARC-H-Direction) compared to the one given by a classic routing (without energy savings) on the germany50 network with different traffic matrices.



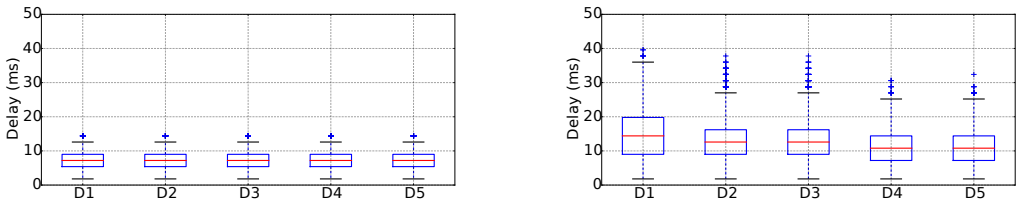
(a) Classic Routing (b) EARC

Figure 11: Average delay by path on the germany50 network



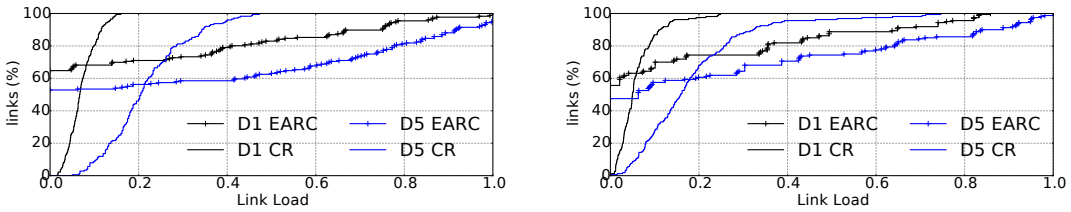
(a) Classic Routing (b) EARC

Figure 12: Average delay by path on the zib54 network.

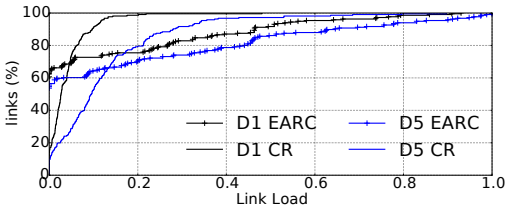


(a) Classic Routing (b) EARC

Figure 13: Average delay by path on the ta2 network.

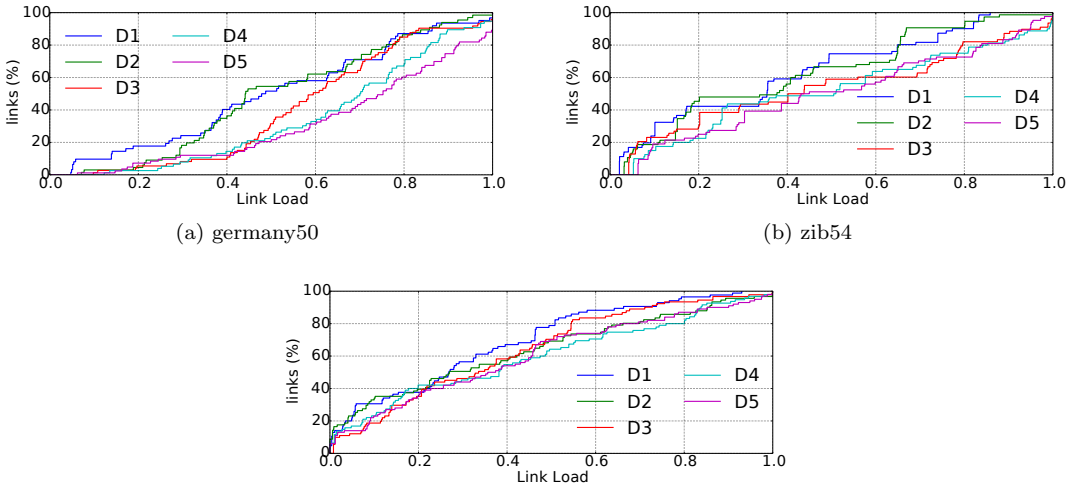


(a) germany50 (b) zib54



(c) ta2

Figure 14: Comparison of the cumulative distribution function of the link load for Energy Aware Routing with Compression (EARC) and classic routing (CR). Results for off peak traffic (D1) and rush hour traffic (D5) are provided.

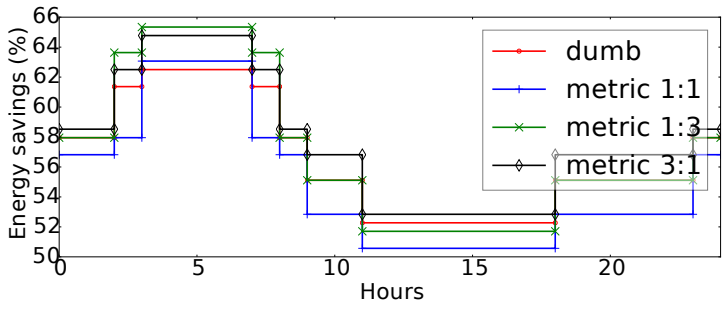


(a) germany50

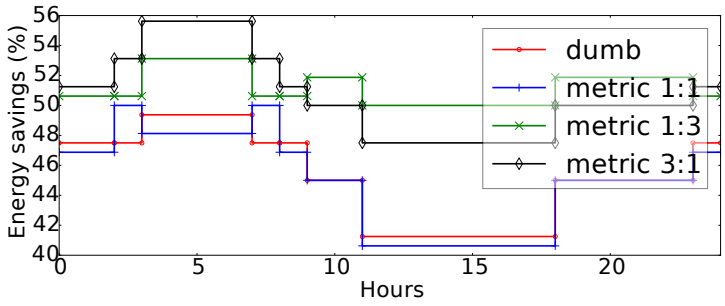
(b) zib54

(c) ta2

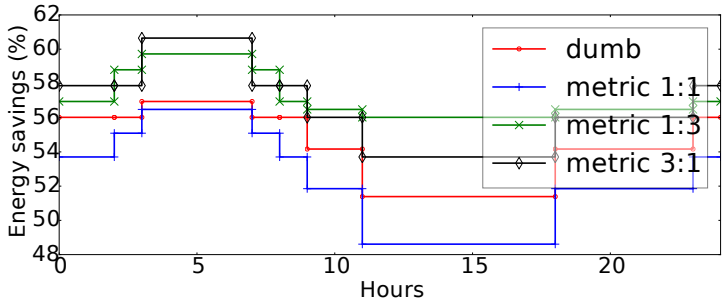
Figure 15: Cumulative distribution function of the link load of the *swichted on links* using EARC for the five demand matrices (D1 is off peak traffic and D5 is rush hour traffic.)



(a) germany50

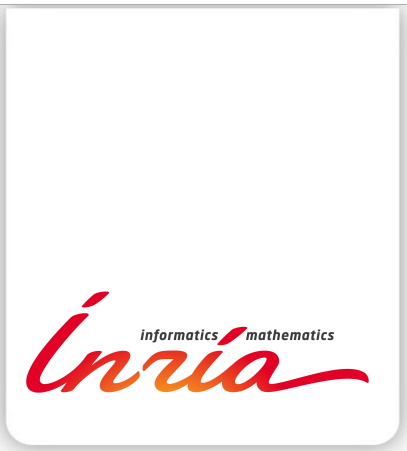


(b) zib54



(c) ta2

Figure 16: Energy savings for the different metrics with EARC-H-Direction. For metric $\alpha : \beta$, α represents the weight of the links and β the weight of the table (See Section 5.2).



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399